Position Paper: Process Isolation and Exascale Operating Systems and Runtimes

Noah Evans
Alcatel–Lucent Bell Laboratories
noah.evans@alcatel-lucent.com

***Abstract*—Exascale Operating Systems and Runtimes need to balance performance and backwards compatibility concerns to meet Exascale challenges. This position paper discusses the use of dynamic linking to allow traditional executables to break process isolation, sacrificing security for performance and compatibility. The primary goal of this method of execution is moving computation to data, either by loading a user process inside the kernel boundary, avoiding the need for boundary crossings in Internode communication, or by moving two processes to the same address space, allowing them to communicate using traditional APIs without the need for new shared memory abstractions.**

I. Introduction

We believe that it is possible to adapt OS abstractions to an Exascale context by changing the underlying execution model, while preserving the interface, preserving expressivity while achieving the necessary performance.

Many of the challenges in developing Exascale Operating Systems and Runtimes (OS/R) systems arise from the need to develop systems that allow the user to combine large (and potentially proprietary)binary code bases while still using the available hardware and network resources in a scalable and efficient manner. Systems must balance:

Performance, ensuring that applications complete in a timely and predictable manner.

Compatibility, allowing users to combine a variety of binary-only libraries and executables from different vendors into new applications.

We observed the interaction of these concerns in the process of developing right weight kernels [1] with minimal noise for the FastOS project.

One of the main sources of these interactions is support for legacy APIs, especially the system call API and file system operations. System calls incur frequent user to kernel transitions and introduce process to process IPC overhead. These overheads lead to large inter and intranode communication latencies.

The traditional way to eliminate these latencies is using OS bypass, where the application contains the device driver, avoiding the user to kernel transitions.

However this approach breaks compatibility with existing APIs and legacy software, requiring users to re-implement application interfaces and OS functionality to support OS bypass.

We are currently exploring an OS/R that we believe that we can provide a traditional API with an underlying such that we can equal OS bypass in performance. It will achieve this performance gain by moving user processes inside the kernel and by grouping multiple user processes into the same address space, trading isolation for performance.

II. Challenges

The current, most popular approach is to either use the traditional operating system stripped down to minimize noise such as Zeptos [2] or by using a Lightweight Kernel providing commodity OS ABI and API support. [3]

However both of these approaches cause problems as paring down full featured operating systems to meet the HPC performance standards lead to a situation where compatibility and performance are compromised.

The approach taken by the NIX [4] and the FusedOS [5] projects aims to provide the appearance and convenient abstractions of an operating system while eliminating many of the underlying inefficiencies.

This leads to a hybrid approach, where the kernel adds new process types to handle core heterogenity and limit OS noise.

A hybrid approach such as this, while providing deterministic execution time and binary compatibility with legacy software, still relies on the traditional process isolation model

, which forces applications although they may be running exclusively on a core, to rely on some form of interrupt or messaging to communicate with the kernel or other processes in order too perform IPC.

The overhead of these superfluous calls leads to large intra– and inter–core latencies as applications are forced to communicate across kernel and process address space boundaries.

III. Our Solution

We are in the process of building a prototype implementation that provides better intra- and inter-node performance by limiting data movement between user/kernel and address space boundaries.

The key observation that underlies this approach is that it is possible to adapt OS abstractions to an Exascale context by changing the underlying execution model, while preserving the interface.

This allows users to continue to use old binary software that relies on commodity APIs while still taking advantage of improved scalability.

We achieve this improvement by allowing the user to trade isolation for performance in running their applications, while preserving traditional OS interfaces.

Our approach aims to decrease inter-node communication latency by moving processes with I/O intensive IPCs into the kernel, eliminating the need for user space pinning of network devices. Devices are accessed directly through the same system call api without causing a context switch.
On the other hand, moving communicating user processes that run on a node to the same address space together means system calls between processes, such as writing on a pipe file descriptor or setting up shared memory are no longer costly.

IV. Remarks /Limitations

By sacrificing process isolation for performance, we give up some of the protections afforded by the ring 0 kernel boundary. While these may not be available, especially on heterogeneous systems. We recognize that another security model is required for Exascale systems base on a similar approach.
One potential solution would be to modify a hardware virtualization system like Palacios [3] to use modern virtualization hardware support, in order to provide a more secure environment for processes crossing boundaries, potentially by using nested page tables to prevent processes operating in kernel memory from accessing all of the kernel memory.
Another solution might be to provide a sandboxed environment which restricts the operations available to processes that are allowed to violate boundaries.
While the implementation of the mechanism that moves processes between boundaries is complete, the support of user space and kernel space system call APIs without isolation is currently still under development.

We are also building a new operating system from scratch, Osprey [6] aimed at predictable cloud computing and HPC environments that incorporates many of these ideas.

V. Conclusions

In this position paper, we described a scheme and prototype implementation of an Exascale OS/R system that allows the user to balance the performance and backwards compatibility of their applications.
With our approach, users can choose to minimize the latency of inter-node communication by allowing processes to operate inside the kernel boundary.
As an alternative, users can minimize the cost of inter-process communication that does not leave a single node by loading and running multiple applications in the same address space. The ability of users to select the appropriate mechanism allows to improve the utilitization of hardware resources.

References

[1] Ronald G. Minnich, Matthew J. Sottile, Sung-Eun Choi, Erik Hendriks and Jim McKie. *Right-Weight Kernels: An Off-the-Shelf Alternative to Custom Light-Weight Kernels.* In ACM SIGOPS Operating Systems Review, 2006, vol. 40, no. 2, pp. 22-28.

[2] ZeptoOS: The small linux for big computers http://www.mcs.anl.gov/research/projects/zeptoos/.

[3] J. Lange, K. Pedretti, T. Hudson, P. Dinda, Z. Cui, L. Xia, P. Bridges, S. Jaconette, M. Levenhagen, R. Brightwell, and P. Widener. *Palacios and Kitten: High performance operating systems for scalable virtualized and native supercomputing.* Technical report, EECS Northwestern University, July 2009.

[4] Francisco J. Ballesteros, Noah Evans, Charles Forsyth, Gorka Guardiola, Jim McKie, Ron Minnich, Enrique Soriano. NIX: A Case for a Manycore System for Cloud Computing. Bell Labs Technical Journal 2012, Vol. 17, No. 2.

[5] Robert W. Wisniewski, Todd Inglett, Yoonho Park, Bryan Rosenburg, Eric Van Hensbergen, Kyung Dong Ryu. *FusedOS: Fusing LWK Performance with FWK Functionality in a Heterogeneous Environment.* Submitted to Supercomputing 2012

[6] Jan Sacha, Jeff Napper, Hening Schild, Sape Mullender, Jim McKie. *Osprey: Operating System for Predictable Clouds.* The Second International Workshop on Dependability of Clouds, Data Centers and Virtual Machine Technology, June 2012.